



**FPT POLYTECHNIC**



[www.poly.edu.vn](http://www.poly.edu.vn)



Conceive Design Implement Operate

## **BASIC SPRING SECURITY**

**GIẢNG VIÊN: NGUYỄN NGHIỆM**

- Web & Spring Security
- Authentication
- Authorization
- Thymeleaf Extras Security





# UNDERSTANDING WEB SECURITY

---



- ❑ Trang web trước và sau khi đăng nhập có thể khác nhau
- ❑ Không thể thực hiện một số hành vi nếu chưa đăng nhập
- ❑ Sau khi đã đăng nhập
  - ❖ Giao diện có thể khác nhau tùy thuộc vào vai trò
  - ❖ Một số chức năng có thể thực hiện được hoặc không tùy vào vai trò
- ❑ Đăng nhập
  - ❖ Từ trang web
  - ❖ Từ mạng xã hội



## ❑ **XSS** – Cross-Site Scripting

- ❖ Ngăn chặn thực hiện của script từ data

## ❑ **CORS** – Cross-Site Resource Sharing

- ❖ Ngăn chặn chia sẻ tài nguyên

## ❑ **CSRF** – Cross-Site Request Forgery

- ❖ Ngăn chặn các request giả lập

## ❑ **Authentication**

- ❖ Ngăn chặn thực hiện khi chưa đăng nhập

## ❑ **Authorization**

- ❖ Ngăn chặn thực hiện khi đăng nhập không đúng vai trò

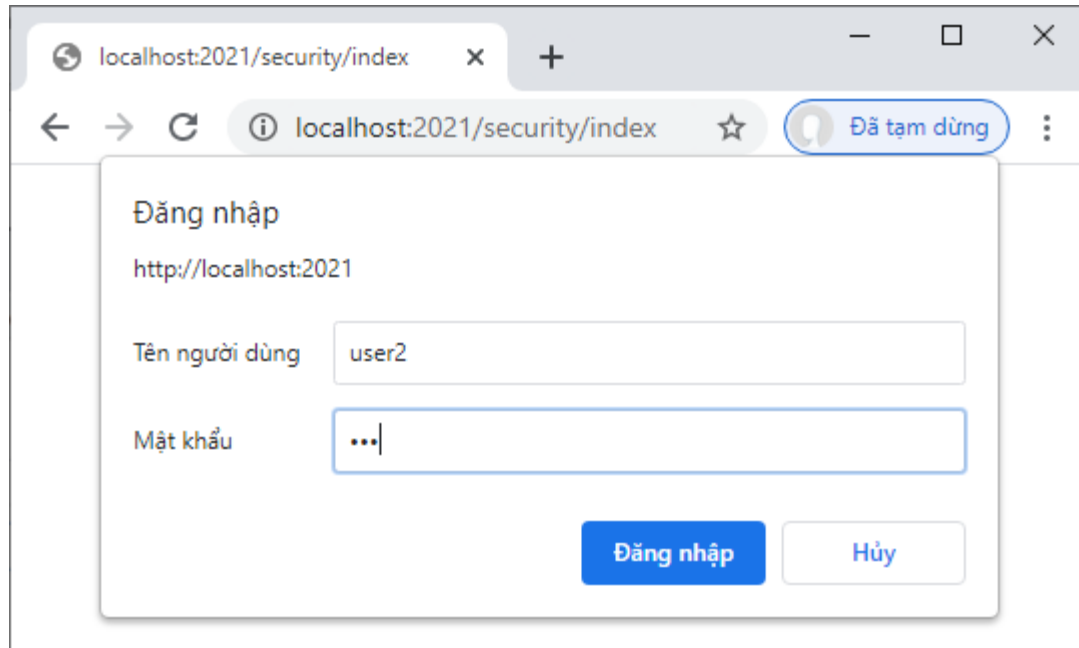
- ❑ Spring Security là một framework, cung cấp các quy chuẩn trong việc thực hiện phòng vệ ứng dụng web
- ❑ Đơn giản trong Validation (XSS), CORS, CSRF
- ❑ Đa dạng hình thức phân quyền sử dụng
  - ❖ Cấu hình
  - ❖ @Annotation
  - ❖ Lập trình
  - ❖ Giao diện
- ❑ Chuẩn hóa mô hình dữ liệu người sử dụng và phương pháp mã hóa mật khẩu
- ❑ Tùy biến hình thức đăng nhập (Login UI), đăng xuất, xử lý lỗi truy cập không đúng vai trò
- ❑ Đăng nhập với user từ mạng xã hội (facebook, gmail...)



# WEB SECURITY CASE STUDY

---





localhost:2021/security/index

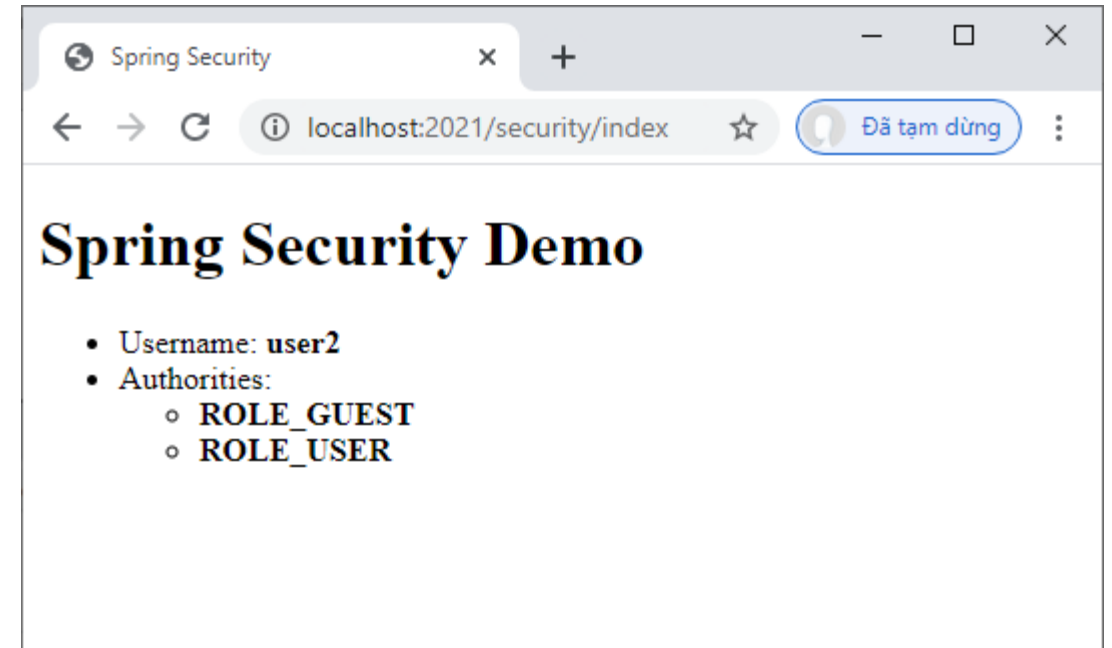
Đăng nhập

http://localhost:2021

Tên người dùng: user2

Mật khẩu: ...

Đăng nhập Hủy



Spring Security

localhost:2021/security/index

## Spring Security Demo

- Username: **user2**
- Authorities:
  - ROLE\_GUEST
  - ROLE\_USER

```
@Controller
public class SecurityController{
    @GetMapping("/security/index")
    public String index() {
        return "index";
    }
}
```

```
<ul th:object="${#authentication.principal}">
    <li>Username: <b>[[*{username}]]</b></li>
    <li>Authorities:
        <ul>
            <li th:each="a : *{authorities}">[[${a}]]</li>
        </ul>
    </li>
</ul>
```



@Configuration

@EnableWebSecurity

**public class** SecurityConfig **extends** WebSecurityConfigurerAdapter {

/\*--MÃ HÓA MẬT KHẨU--\*/

@Bean

**public** BCryptPasswordEncoder getPasswordEncoder() {

**return new** BCryptPasswordEncoder();

}

/\*--QUẢN LÝ NGƯỜI DỮ LIỆU NGƯỜI SỬ DỤNG--\*/

@Override

**protected void** configure(AuthenticationManagerBuilder auth) **throws** Exception {...}

/\*--PHÂN QUYỀN SỬ DỤNG VÀ HÌNH THỨC ĐĂNG NHẬP--\*/

@Override

**protected void** configure(HttpSecurity http) **throws** Exception {...}

}

```
/*--QUẢN LÝ NGUỒN DỮ LIỆU NGƯỜI SỬ DỤNG--*/
```

```
@Override
```

```
protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
    BCryptPasswordEncoder pe = new BCryptPasswordEncoder();  
  
    auth.inMemoryAuthentication()  
        .withUser("user1").password(pe.encode("123")).roles("GUEST")  
        .and()  
        .withUser("user2").password(pe.encode("123")).roles("USER", "GUEST")  
        .and()  
        .withUser("user3").password(pe.encode("123")).roles("USER", "GUEST", "ADMIN");  
}
```

### ❑ inMemoryAuthentication()

- ❖ Quản lý nguồn dữ liệu người sử dụng trong bộ nhớ
- ❖ Cơ chế mã hóa mật khẩu
- ❖ Mỗi user có thể có nhiều vai trò

/\*--PHÂN QUYỀN SỬ DỤNG VÀ HÌNH THỨC ĐĂNG NHẬP--\*/

@Override

```
protected void configure(HttpSecurity http) throws Exception {  
    // CSRF, CORS  
    http.csrf().disable().cors().disable();  
    // Phân quyền sử dụng  
    http.authorizeRequests()  
        .anyRequest().authenticated();  
    // Giao diện đăng nhập  
    http.httpBasic();  
}
```

Bắt buộc đăng nhập khi truy xuất bất kỳ URL nào

Giao diện đăng nhập là hộp thoại của trình duyệt

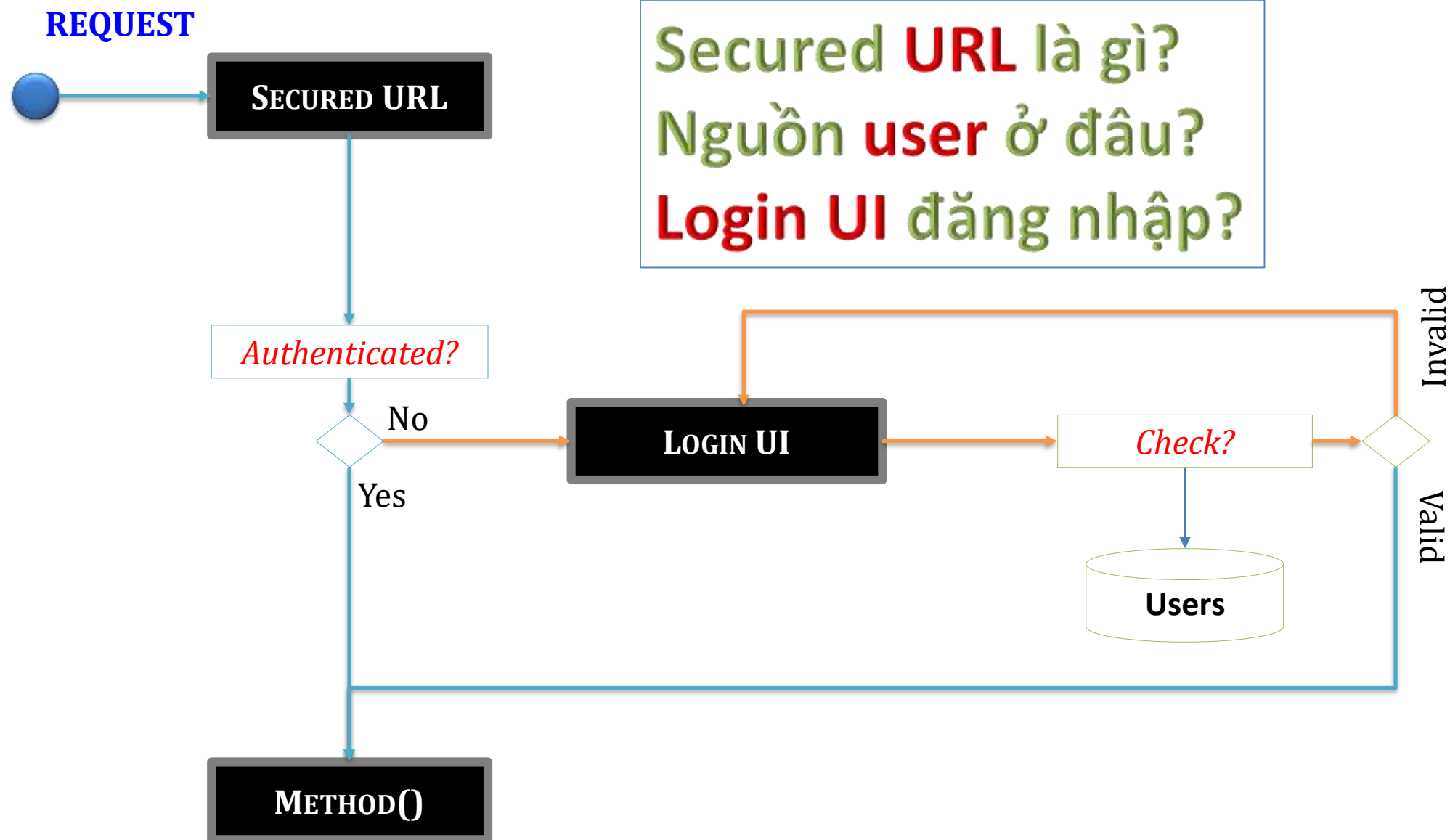


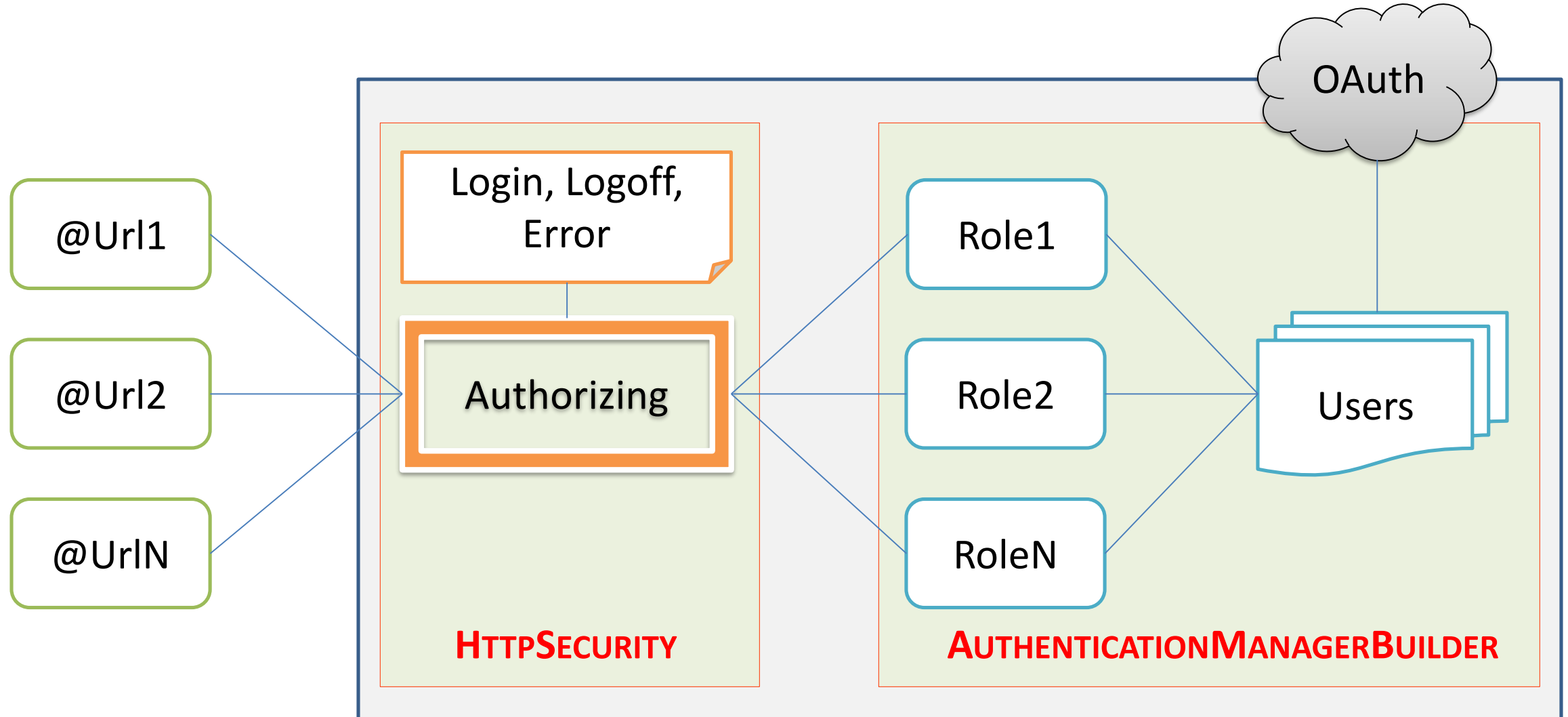
{  
[

}  
]

**DEMO**





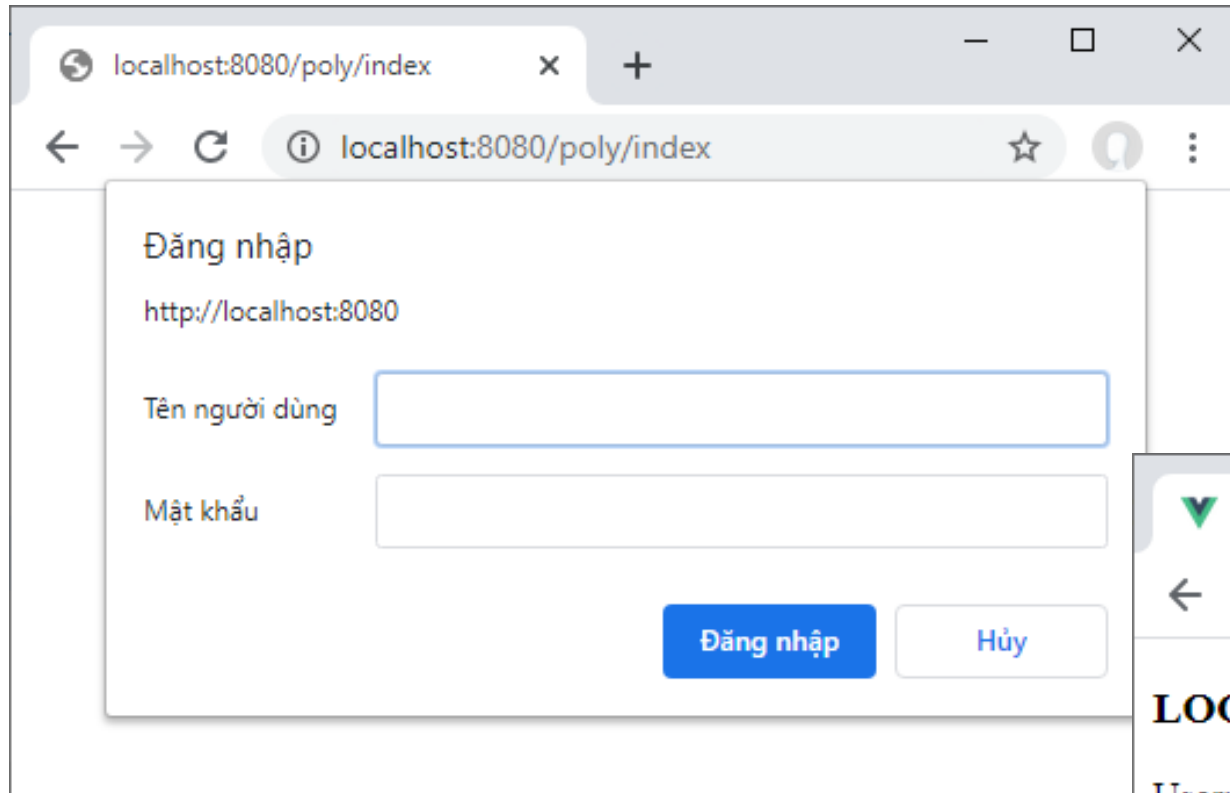




## HTTPSECURITY – LOGIN & LOGOFF

---

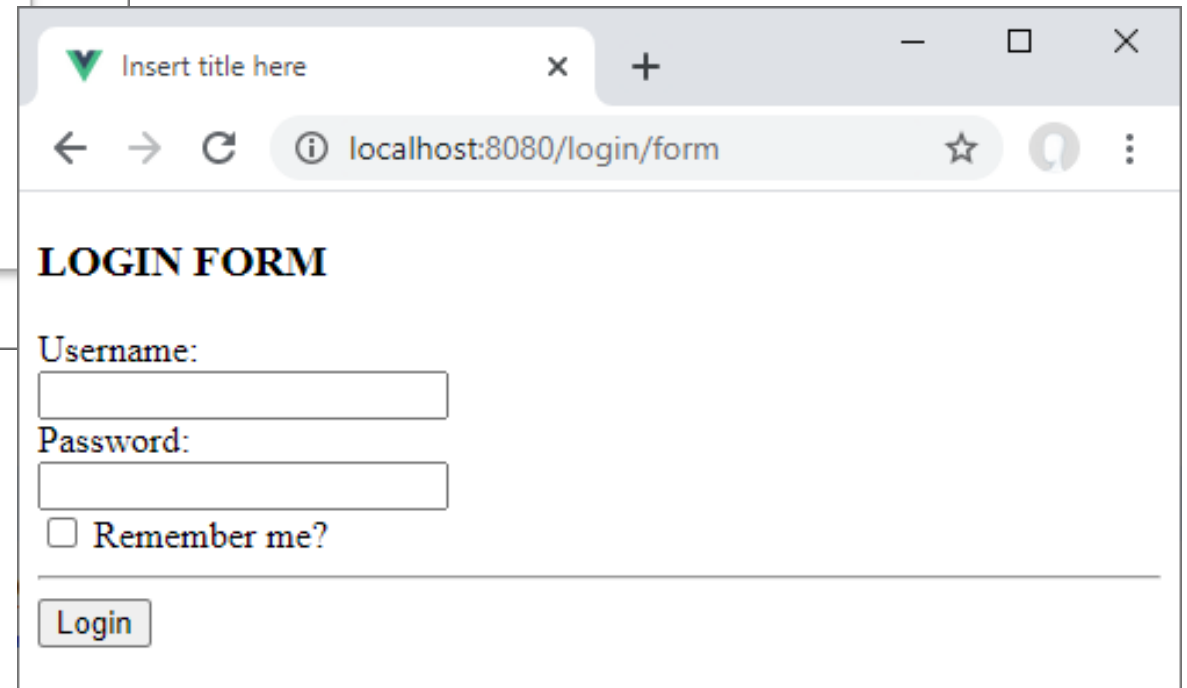




A screenshot of a web browser window with the address bar showing `localhost:8080/poly/index`. A modal dialog box titled "Đăng nhập" (Login) is displayed. The dialog shows the base URL `http://localhost:8080`. It contains two input fields: "Tên người dùng" (Username) and "Mật khẩu" (Password). At the bottom right of the dialog are two buttons: "Đăng nhập" (Login) in blue and "Hủy" (Cancel) in white.

`http.httpBasic()`

`http.formLogin()`



A screenshot of a web browser window with the address bar showing `localhost:8080/login/form`. The page content is titled "LOGIN FORM". It contains the following elements:

- Label "Username:" followed by a text input field.
- Label "Password:" followed by a text input field.
- A checkbox labeled "Remember me?".
- A "Login" button at the bottom.

□ Thay thế hộp thoại đăng nhập bằng form HTML

```
<a href="/security/logoff">Sign Out</a>
```

```
<form action="/security/login" method="post">  
  <input name="username" placeholder="Username?">  
  <br>  
  <input name="password" placeholder="Password?">  
  <br>  
  <input type="checkbox" name="remember"> Remember me?  
  <hr>  
  <button>Login</button>  
</form>
```

// GIAO DIỆN ĐĂNG NHẬP &amp; ĐĂNG XUẤT

```
http.formLogin()  
    .loginPage("/security/form")  
    .loginProcessingUrl("/security/login")  
    .defaultSuccessUrl("/security/index", false)  
    .failureUrl("/security/form?error=true")  
    .usernameParameter("username")  
    .passwordParameter("password");  
http.rememberMe()  
    .rememberMeParameter("remember");  
http.logout()  
    .logoutUrl("/security/logoff")  
    .logoutSuccessUrl("/security/index")  
    .addLogoutHandler(new SecurityContextLogoutHandler());
```

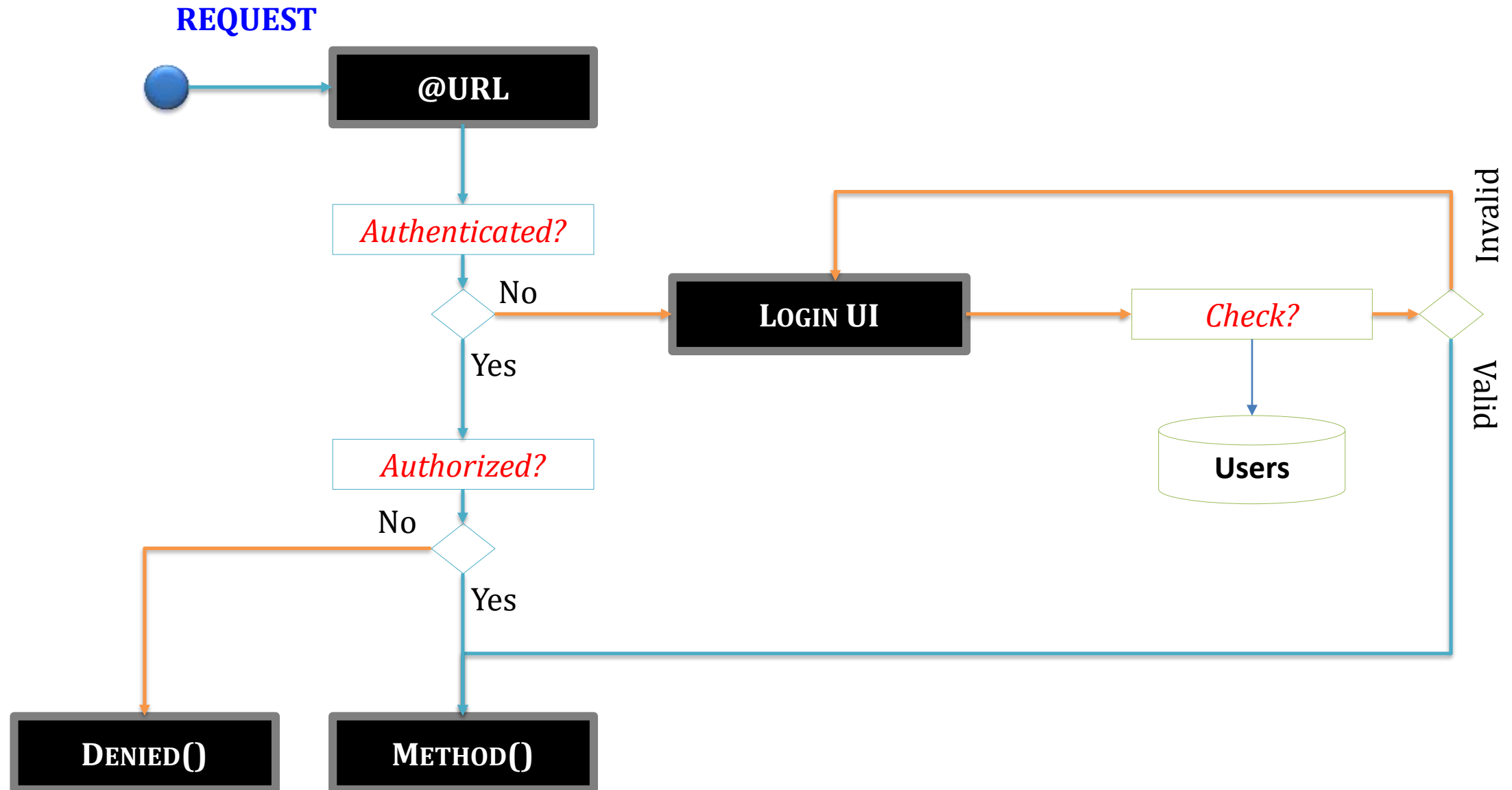
`http.httpBasic()`

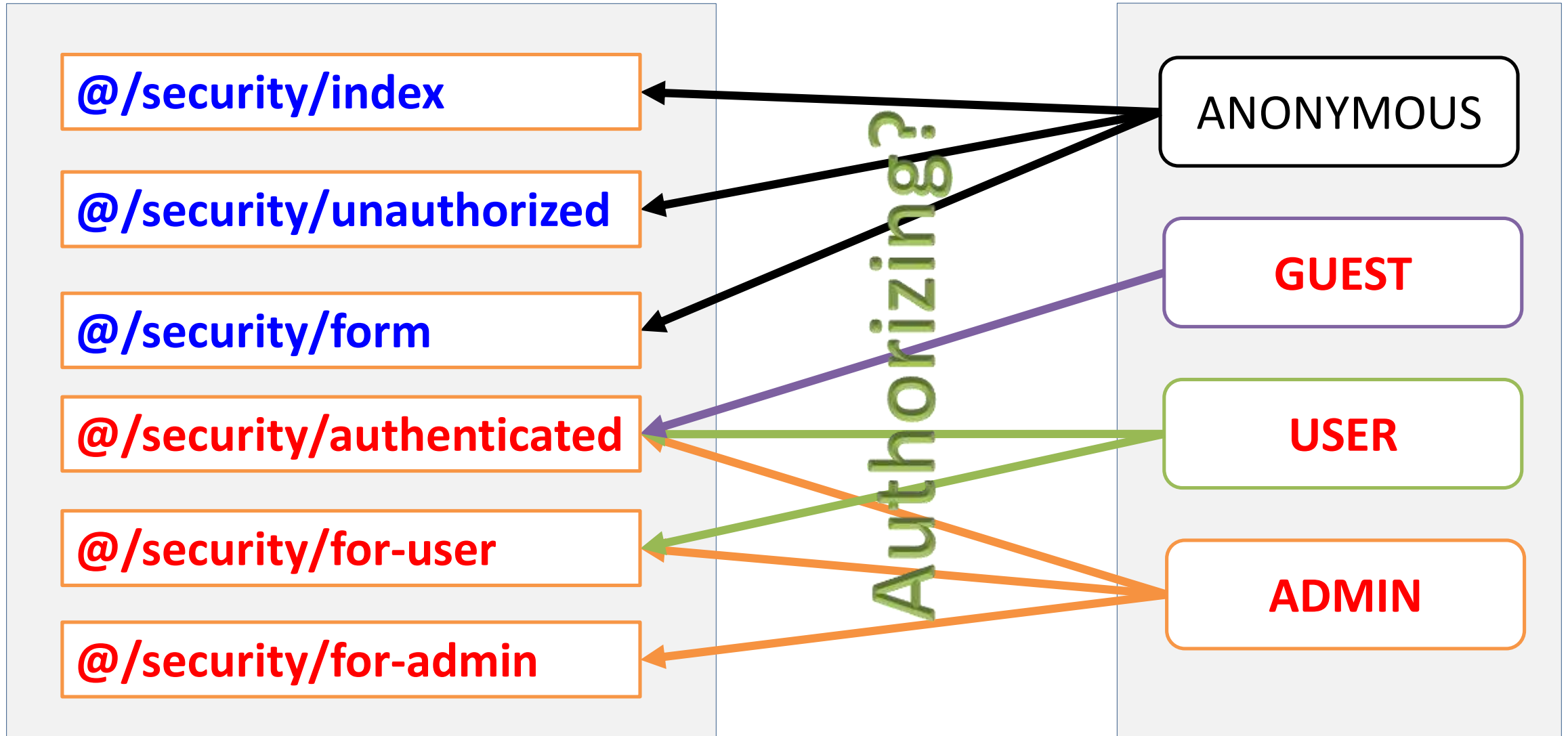
```
<form action="/security/login">  
  <input name="username">  
  <input name="password">  
  <input name="remember"  
    type="checkbox">  
  <a href="/security/logoff">
```



## HTTPSECURITY - AUTHORIZING

---





## Authorizing

1

Configuration

hasRole()  
hasAnyRole()  
authenticated()  
permitAll()

URL

2

@PreAuthorize

hasRole()  
hasAnyRole()  
isAuthenticated()

URL

3

Programming

Authentication  
HttpServletRequest

Code Fragment

4

Thymeleaf

hasRole()  
hasAnyRole()  
isAuthenticated()

UI Fragment



# 1. AUTHORIZING WITH CONFIGURATION

// PHÂN QUYỀN SỬ DỤNG

```
http.authorizeRequests()  
    .antMatchers("/security/for-admin").hasRole("ADMIN")  
    .antMatchers("/security/for-user").hasAnyRole("ADMIN", "USER")  
    .antMatchers("/security/authenticated").hasAnyRole("ADMIN", "USER", "GUEST")  
    .anyRequest().permitAll();
```

// ĐIỀU KHIỂN LỖI TRUY CẬP KHÔNG ĐÚNG VAI TRÒ

```
http.exceptionHandling().accessDeniedPage("/security/unauthorized");
```

## 1. XÁC ĐỊNH URLS

- ❖ *antMatchers*(String...urlPatterns)
- ❖ *anyRequest*()

## 2. PHÂN QUYỀN

- ❖ *hasRole*(String role)
- ❖ *hasAnyRole*(String...roles)
- ❖ *permitAll*()
- ❖ *authenticated*()

## 2.1. AUTHORIZING USING @PREAUTHORIZE()

```
@PreAuthorize("isAuthenticated()")  
@RequestMapping("/security/authenticated")
```

```
@PreAuthorize("hasAnyRole('ADMIN', 'USER')")  
@RequestMapping("/security/for-user ")
```

```
@PreAuthorize("hasRole('ADMIN')")  
@RequestMapping("/security/for-admin ")
```

## 2.2. AUTHORIZING USING @PREAUTHORIZE()

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = **true**)

**public class** SecurityConfig **extends** WebSecurityConfigurerAdapter{

@Override

**protected void** configure(HttpSecurity http) **throws** Exception {

...

http.authorizeRequests()

.anyRequest().**permitAll**();

...

}

...

}

Cho phép sử dụng Annotation  
trong @Controller

Có thể không cần cấu hình bảo  
mật ở đây

### 3. AUTHORIZING BY PROGRAMMING

```
@RequestMapping("/page")
public String prog(Authentication auth, HttpServletRequest req) {
    if(auth.isAuthenticated()){
        String username = auth.getName();
        Collection<?> authorities = auth.getAuthorities();
        UserDetails user = (UserDetails) auth.getPrincipal();
        if(req.isUserInRole("GUEST")) {...}
        else {...}
    }
    else {...}
    ...
}
```

#### ❑ Authentication API

- ❖ *isAuthenticated()*: Boolean
- ❖ *getName()*: String
- ❖ *getAuthorities()*: List<GrantedAuthority>
- ❖ *getPrincipal()*: **UserDetails**

## 4.1. AUTHORIZING WITH THYMELEAF

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<body>
    <ul>
        <li th:text="${#authentication.name}"></li>
        <li th:text="${#authentication.authorities}"></li>
    </ul>
    <ul th:if="${#authentication.authenticated}">
        <li th:if="${#request.isUserInRole('ADMIN')}">Admin</li>
        <li th:if="${#request.isUserInRole('ADMIN')} OR ${#request.isUserInRole('USER')}">
            Admin or User
        </li>
        <li th:if="${#request.isUserInRole('GUEST')}">Guest</li>
    </ul>
</body>
</html>
```

## 4.2. AUTHORIZING WITH THYMELEAF-EXTRAS

```
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity5">
<body>
  <ul>
    <li sec:authentication="name"></li>
    <li sec:authentication="authorities"></li>
  </ul>
  <ul sec:authorize="isAuthenticated()">
    <li sec:authorize="hasRole('ADMIN')>Admin</li>
    <li sec:authorize="hasAnyRole('ADMIN', 'USER')>Admin or User</li>
    <li sec:authorize="hasRole('GUEST')>Guest</li>
  </ul>
</body>
</html>
```

```
<dependency>
  <groupId>org.thymeleaf.extras</groupId>
  <artifactId>thymeleaf-extras-springsecurity5</artifactId>
</dependency>
```

{  
[

}  
]

**DEMO**





- ✓ Web Security
  - ✓ XSS, CORS, CSRF, Authorization, Authentication
- ✓ Spring Security
  - ✓ AuthenticationManagerBuilder
  - ✓ Authorizing
- ✓ Customizing Login UI
  - ✓ HttpBasic, HTML Form
- ✓ Authorizing
  - ✓ Configuration
  - ✓ @PreAuthorize
  - ✓ Programming
  - ✓ Thymeleaf Security





**Cảm ơn**